Kauno technologijos universitetas

Informatikos fakultetas

Paprastųjų diferencialinių lygčių sprendimas

P170B115 Skaitiniai metodai ir algoritmai

Ketvirtas laboratorinis darbas

Projekto autorius
Gustas Klevinskas

Akademinė grupė
IFF-8/7

Vadovas
Andrius Kriščiūnas

Kaunas, 2020

Turinys

Įvadas

Užduoties variantas – 15.

m masės sviedinys iššaunamas vertikaliai į viršų pradiniu greičiu v_0 iš aukščio h_0. Žinoma, kad oro pasipriešinimas proporcingas sviedinio greičio kvadratui, o proporcingumo koeficientas lygus k_1, kai sviedinys kyla, ir k_2, kai sviedinys leidžiasi. Kokį maksimalų aukštį ir kada pasieks sviedinys? Kada sviedinys nusileis ant žemės?

1 lentelė. 15 varianto duomenys.

m, kg	v_0, m/s	h_0, m	k_1, kg/m	k_2, kg/m
5	80	5	0.15	0.6

Diferencialinės lygties sudarymas

Surašius visas jėgas, veikiančias sviedinį, gaunama (1) lygybė.

$$m\frac{dv}{dt} = kv^2 + mg \tag{1}$$

Kairėje lygybės pusėje yra aprašoma sviedinį veikiantį jėga, dešinėje pusėje atitinkamai kūną veikiantis oro pasipriešinimas ir sunkio jėga.

Pertvarkius (1) lygybę ir pakeitus greičio išvestinę į aukščio išvestinę gaunama (2) lygybė.

$$\frac{d^2h}{dt^2} = \frac{k}{m}\left(\frac{dh}{dt}\right)^2 + g \tag{2}$$

Iš (2) lygybės gaunama pirmos eilės diferencialinė lygtis pavaizduota (3) formulėje kartu su pradiniais lygčių sistemos sprendiniais.

$$\frac{d}{dt}\begin{Bmatrix} h \\ v \end{Bmatrix} = \begin{Bmatrix} v \\ \frac{k}{m} \cdot v^2 + g \end{Bmatrix}, \qquad \begin{matrix} h_0 = 5 \\ v_0 = 80 \end{matrix} \tag{3}$$

Lygčių sistemos sprendimas

Skaičiavimus atlikau su 0.1, 0.05 ir 0.01 dydžio žingsniais. Rezultatai pateikti lentelėse 2 ir 3.

2 lentelė. Rezultatai skaičiuojant Eulerio metodu.

Žingsnis	Maksimalus aukštis, m	Maksimalaus aukščio pasiekimo laikas, s	Nusileidimo laikas, s
0.1	45.51	2.30	8.00
0.05	50.56	2.40	8.65
0.01	54.46	2.47	9.13

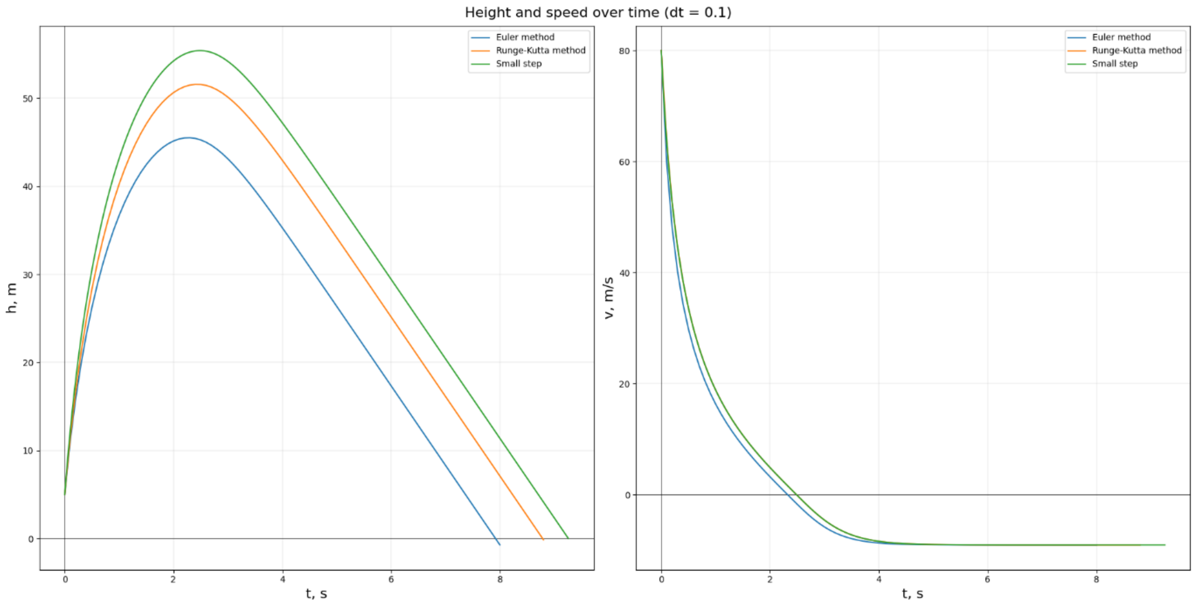
3 lentelė. Rezultatai skaičiuojant Rungės ir Kutos metodu.

Žingsnis	Maksimalus aukštis, m	Maksimalaus aukščio pasiekimo laikas, s	Nusileidimo laikas, s
0.1	51.58	2.40	8.80
0.05	53.46	2.45	9.05
0.01	55.02	2.48	9.21

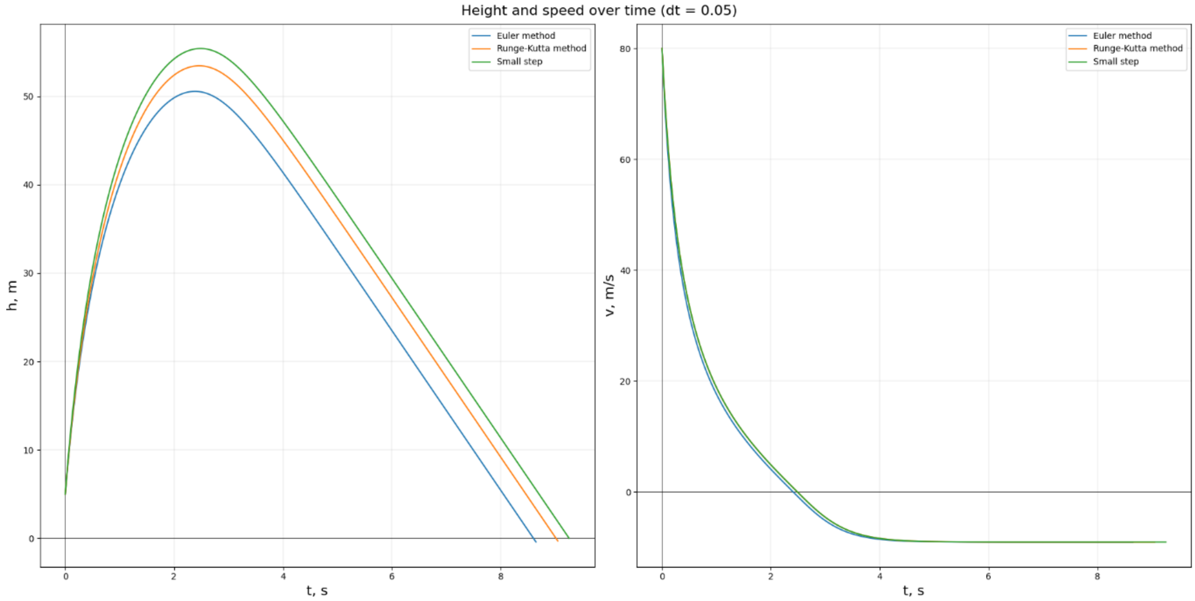
4 lentelė. Rezultatai gauti naudojant labai mažą žingsnį.

Žingsnis	Maksimalus aukštis, m	Maksimalaus aukščio pasiekimo laikas, s	Nusileidimo laikas, s
0.0001	55.41	2.49	9.26

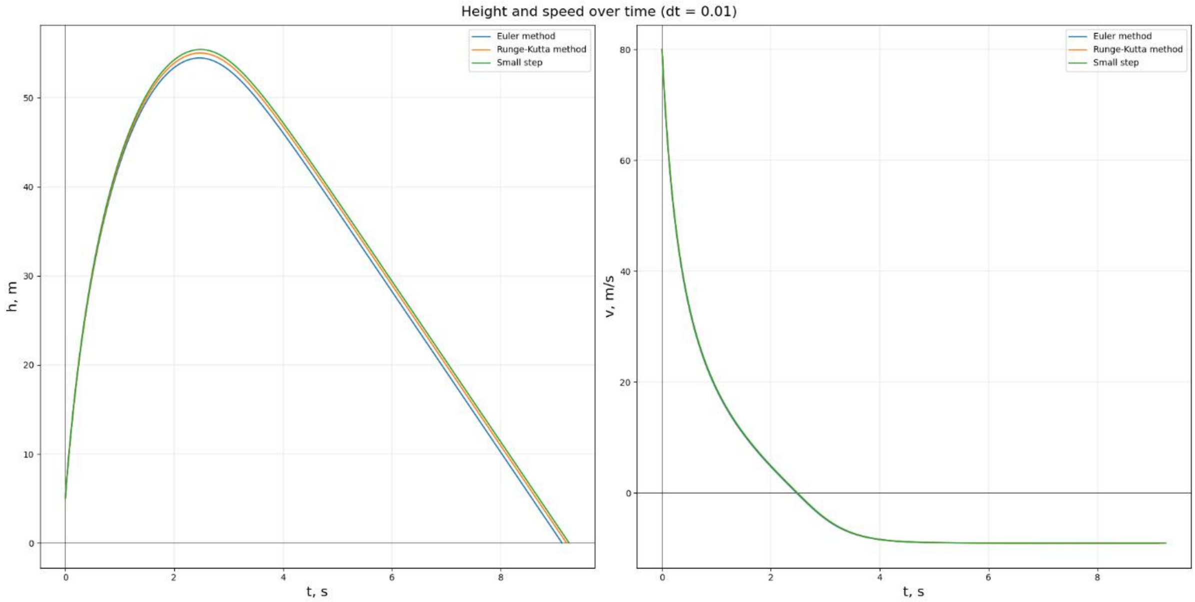
4 lentelėje pavaizduoti rezultatai naudojant mažą žingsnį su Rungės ir Kutos metodu. Dėl šio mažo žingsnio rezultatai yra labai arti tikrų. Taip galima patikrinti 2 ir 3 lentelėse gautų rezultatų teisingumą.



1 pav. Aukščio ir greičio grafikai pagal laiką naudojant 0.1 žingsnį.



2 pav. Aukščio ir greičio grafikai pagal laiką naudojant 0.05 žingsnį.



3 pav. Aukščio ir greičio grafikai pagal laiką naudojant 0.01 žingsnį.

Tiek iš 1 – 3 pav., tiek iš rezultatų 2 – 4 lentelėse matoma, jog Rungės-Kuto metodas yra kur kas tikslesnis lyginant su Eulerio metodu, tačiau didėjant žingsnių skaičiui abiejų metodų rezultatai konverguoja į „teisingą“. Rungės-Kuto metodas dėl to leidžia naudoti mažesnį žingsnių skaičių ir greičiau apskaičiuoti rezultatą.

Rezultatų lyginimas su kitais šaltiniais

Gautus rezultatus bandžiau pasitikrinti naudojant Python biblioteka *scipy*. Tačiau kilo keblumų: bibliotekos *odeint* funkcijai reikia perduoti funkciją, apskaičiuojančią $\frac{dy}{dx}$ dalį bendroje funkcijoje $y_{n+1} = y_n + \Delta x \cdot \frac{dy}{dx}$. Bet uždavinyje ji priklauso nuo sąlyginių kintamųjų k_1 ir k_2, kurie patys priklauso nuo y (arba greičio anksčiau minėtose formulėse).

Šią problemą bandžiau išspręsti iš pradžių apskaičiuojant su k_1, randant tašką, kada greitis pasikeičia į neigiamą, atmetant tolimesnius rezultatus, apskaičiavus aukštį tame taške ir vėl paleidus *odeint* funkciją su apskaičiuotu aukščiu kaip pradiniu argumentu ir k_2 vietoj k_1. Bet kilo bėdų su pirmo paleidimo rezultatų ir antro rezultatų apjungimu.

Dar bandžiau naudotis interneto svetainėmis www.symbolab.com ir www.wolframalpha.com. Tačiau iš jų taip pat nepavyko gauti rezultato.

Programinis kodas

```
import matplotlib.pyplot as plt
import numpy as np

m = 5
v0 = 80
h0 = 5
k1 = 0.15
k2 = 0.6
g = 9.8

dt = 0.1


def f(v, k):
    return k / m * v * abs(v) + g


def euler_method(v, k):
    return dt * f(v, k)


def runge_kutta_method(v, k):
    v1 = v - dt / 2 * f(v, k)
    v2 = v - dt / 2 * f(v1, k)
    v3 = v - dt * f(v2, k)

    return dt / 6 * (f(v, k) + 2 * f(v1, k) + 2 * f(v2, k) + f(v3, k))


def solve(method):
    t_arr = [0]
    h_arr = [h0]
    v_arr = [v0]

    t = 0
    h = h0
    v = v0
    k = k1

    while h > 0:
        prev_v = v

        v -= method(v, k)
        h += dt * v
        t += dt

        v_arr.append(v)
        h_arr.append(h)
        t_arr.append(t)

        if np.sign(prev_v) != np.sign(v):
            k = k2

    return t_arr, h_arr, v_arr


def plot(axis, y_label, *plots):
    axis.axvline(linewidth=0.5, color='k')
    axis.axhline(linewidth=0.5, color='k')
    axis.grid(linewidth=0.2)
    axis.set_xlabel('t, s', fontsize=16)
    axis.set_ylabel(y_label, fontsize=16)

    lines = []
    names = []

    for i in plots:
        line, = axis.plot(i[0], i[1])
        lines.append(line)
        names.append(i[2])
```

```
    axis.legend(lines, names)


def print_result(title, t_arr, h_arr):
    print('----- {} -----'.format(title))
    max_h = max(h_arr)
    time = t_arr[h_arr.index(max_h)]
    print('Max height = {:.2f} m, time = {:.2f} s'.format(max_h, time))
    print('Landing time = {:.2f} s'.format(t_arr[-1]))


fig, axs = plt.subplots(1, 2)
fig.suptitle('Height and speed over time (dt = {})'.format(dt), fontsize=16)

euler_time, euler_h, euler_v = solve(euler_method)
runge_time, runge_h, runge_v = solve(runge_kutta_method)

dt = 0.0001
true_time, true_h, true_v = solve(runge_kutta_method)

plot(
    axs[0],
    'h, m',
    (euler_time, euler_h, 'Euler method'),
    (runge_time, runge_h, 'Runge-Kutta method'),
    (true_time, true_h, 'Small step'),
)
plot(
    axs[1],
    'v, m/s',
    (euler_time, euler_v, 'Euler method'),
    (runge_time, runge_v, 'Runge-Kutta method'),
    (true_time, true_v, 'Small step'),
)

print_result('Euler method', euler_time, euler_h)
print_result('Runge-Kutta method', runge_time, runge_h)
print_result('Runge-Kutta with dt = {}'.format(dt), true_time, true_h)

plt.show()
```